



LTC®6915
Best Instrumentation Amp
-Now Digitally Controlled



What happened to my serial port? Build a USB-to-serial port converter

By Dan Harmon, USB/1394 Catalog Products Marketing Manager, Texas Instruments, Dallas

planetanalog

Nov 18, 2003 (5:52 PM)

URL: <http://www.planetanalog.com/showArticle?articleID=16101224>

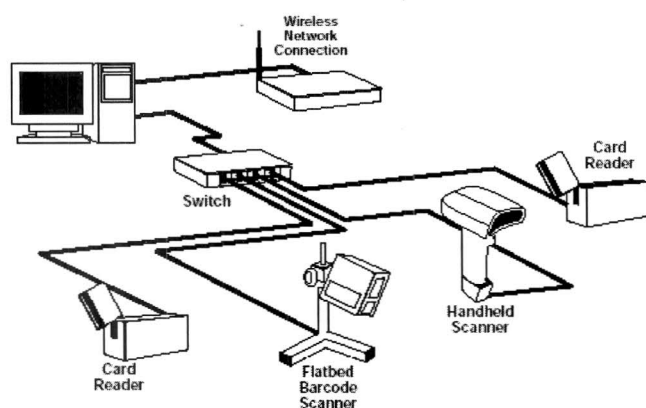
For years now, we have been hearing about the benefits that legacy-free PCs are going to deliver. The average user wants less confusion in their PC as they try and connect additional new peripheral devices. One question has haunted them for years: "Which device plugs in where?" By removing the older interface ports " serial, parallel, PS2 " and replacing them with plug-and-play ports such as USB, the end-user should experience dramatic improvement.

USB delivers easy expansion to the consumer. Many devices have native class driver support in the operating system and do not require drivers to be supplied by the peripheral developer, making it even simpler to add these peripherals. The ability to expand the bus via the use of hubs also has made adding peripherals much easier for the average user. Legacy-free PCs are now a reality, offering these benefits to the average user.

Embedded applications/instruments don't want to change!

Many industrial, professional or embedded applications, however, are not as excited about legacy-free PCs as the average user. Many of these applications have been using the various UART interfaces and neither need nor want to make the transition. Whether RS-232, RS-422 or RS-485, the UART connection has been the mainstay in lower bandwidth communications for decades. For control, monitoring and low volume data transfer, UART connections have provided a low-cost, easy-to-use solution. The system developers for these applications have spent countless hours and dollars developing these systems and are satisfied with the performance.

Figure 1: Typical Existing Industrial Serial Interface Application.



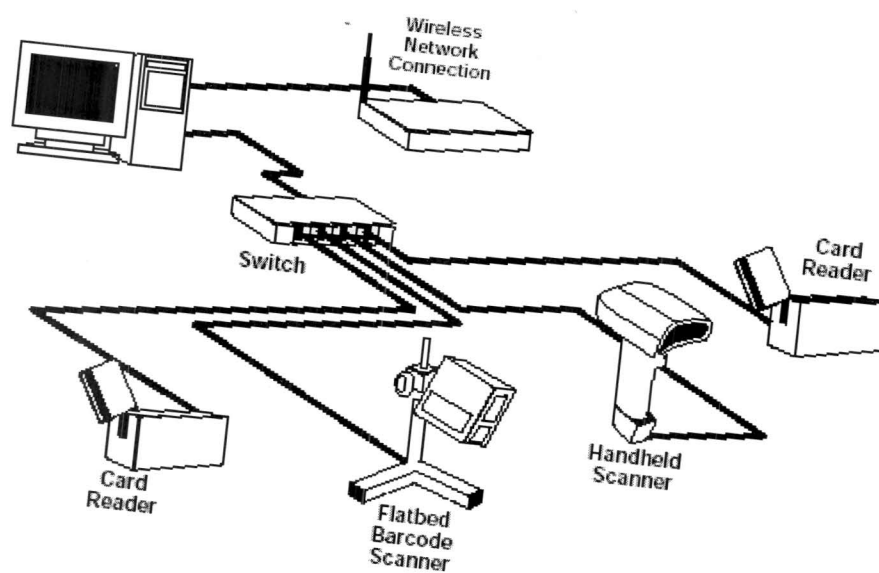
The system-level design and implementation for these customized application-specific products has created a system architecture that has been stable for many years and provides all the needed functionality and performance. The firmware developed for the system processor to implement the needed functions has assumed the data being transferred is utilizing the UART connection. In addition, the host application software has been developed and optimized throughout the lifecycle of these products and it, too, assumes a UART connection. Changing any of these items would require a significant investment. The benefits of a legacy-free PC are not as evident to these professional users.

There are three basic methods these developers can use to adapt their system to a legacy-free PC. The first would be a complete system redesign to natively support a USB connection. The second would be to use a commercially available USB-to-RS-232 adapter. A third method would be to use a USB-to-UART adapter customized for the system application. Let's look at some of the advantages and disadvantages of each of the three alternatives.

The designer's alternatives

The complete system redesign alternative would involve one or more of the following efforts. It most likely would require a new system processor/microcontroller. Depending on the ports available on the current processor, the system designer would need to either convert to a new controller that has native USB support or convert the connection to a new port on the processor and host PC and use an alternative bridging solution. Examples of alternative bridging solutions would include a memory-mapped interface or a parallel host processor interface. One of the main benefits of switching to a native USB solution would be a potential throughput improvement.

In addition to these hardware changes, software changes would also be necessary. The change to a new processor will most likely also require new device firmware, since the processor/microcontroller firmware needs to account for the new method of sending and receiving data. There will also be a need to modify the host user application software to address this new connection methodology. This software currently interfaces to the serial port and will now need to interface directly to the new connection port. In addition to the application software, the host driver will need to be modified for the change in port connection" whether USB or other. If the application can be tailored to one of the native USB class drivers (see www.usb.org for more information on class drivers), this will allow the peripheral OEM to not ship an installation disk with their device and potentially allow it to be plug-and-play compatible with any of the new PCs if specific user application software is not required for basic functionality.



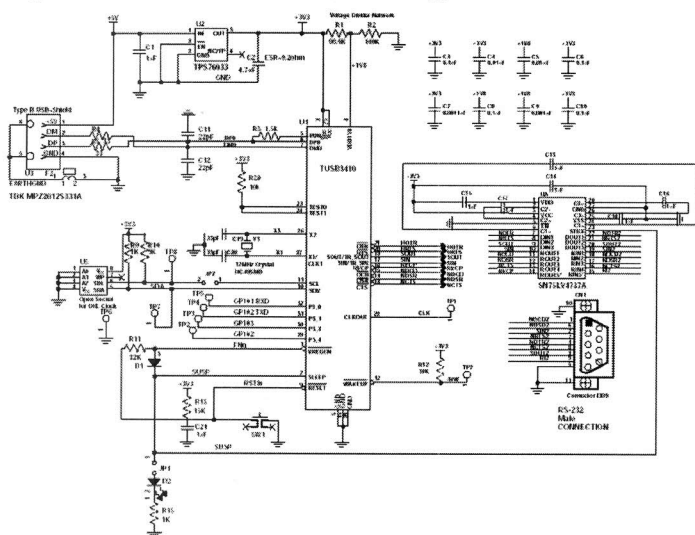
The second alternative is to use one of the many commercially available USB-to-serial converter dongles/cables. There are many different suppliers of these devices. These devices would quickly enable connection to the legacy"free PC, creating a virtual COM port (VCP). Assuming the application software can map to any COM port, it should be a simple matter of remapping the application to talk to the new VCP. A VCP driver is supplied for each of these dongles and must be installed on the PC to enable the port. Many of these dongles have been tailored to specific applications such as PDA cradles or serial MODEMs. While this alternative should get you to market the quickest, there are some issues that the designer will need to assess the impact on their overall product strategy. Using an off-the-shelf product will limit the designer's ability to control the compatibility, quality or branding of the accessory that is used with their product, potentially leading to an increase in service calls and negatively impacting the customer's evaluation of the product. Logistical issues also need to be considered related to the ability to control cost or on-going supply of the selected dongle. Lastly, this will be a fairly costly implementation on a per unit basis.

The third alternative is to design your own application-specific USB-to-serial dongle or embedded bridge. This has most of the benefits of using the off-the-shelf alternative and addresses many of its shortcomings. This alternative will allow your bridge to appear as a VCP to both the external system and to the host application software. It also will require a VCP driver, but it can be tailored to your specific application. Building your own dongle/embedded bridge will provide control over compatibility, quality, branding, cost and supply. The embedded bridge version would require a board change, but will have a lower BOM cost than the external dongle. Let's look at these two approaches in a little more detail.

The Application"Specific External Dongle

The application-specific external dongle solution is very much like purchasing an off-the-shelf dongle, but it allows you to address all of the shortcomings of that approach. The existing system implementation requires no changes, and the hardware requires no modification. You would simply hook the serial interface of the dongle up to the existing serial port of your equipment. By designing your own dongle, you can also optimize the line driver or transceiver for your specific application need: RS-232, RS-422, RS-485, LVDS, etc. Most commercially available dongles are limited to RS-232 and will not work for the other serial interfaces.

Figure 2: RS-232 External Dongle Reference Schematic.



Likewise, the host application software should not require any changes. The only potential issue is if your application can map to different COM ports or if it is permanently mapped to a specific COM port. If it is permanently mapped to a specific COM port, it is recommended that this software be changed to allow for COM port mapping by the end-user. Since this approach does create a new VCP on the computer, a VCP driver will need to be shipped with your product. By designing and building your own dongle, as opposed to using a commercially available dongle, you have the ability to customize your driver to your specific application needs. This will allow for better functionality and compatibility between your end-equipment and the dongle.

An additional benefit of this approach is the ability for one end-equipment "box" to be used with both legacy-capable PCs as well as the newer legacy-free PCs. The biggest drawback to this approach is the overall implementation cost. When you amortize the development cost on a per unit basis and add it to the actual BOM cost of an external dongle, this is most likely the highest overall implementation cost.

The Embedded Bridge "Dongle"

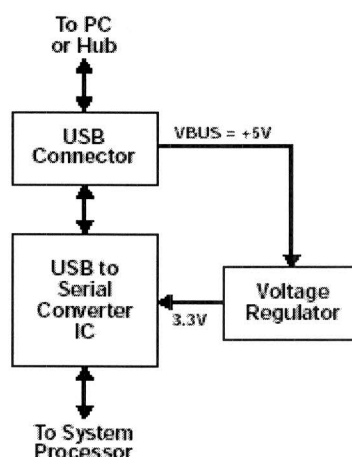


Figure 3: Embedded Bridge System Block Diagram.

The embedded bridge approach is to take the "dongle" and place it on board in the end-equipment. It is essentially a board-level implementation of the external dongle. By embedding the USB-to-serial bridge into the "box", you allow it to look like a native USB device, even if it ultimately it is still a VCP connection.

Since to the system processor and the host PC, this appears as a VCP just like in the application-specific external dongle case, the software changes needed would be minimal. Any required changes should be identical to those needed in the application-specific external dongle.

By placing the bridge device on board, you can save significant per-unit cost. You remove the cost of the miscellaneous structure (i.e. printed circuit board, case or housing, connector hardware, etc.) associated with an external dongle accessory. In addition, you can save on the required electronics by placing the bridge device on board. Any serial cable transceivers (RS-232, RS-422, RS-485, etc.) are no longer required between the bridge device and the system processor/microcontroller. Obviously, the bridge device on board with the system processor would require a board layout change. Even with the development cost, when you amortize this over the lifetime of your product, this will most likely be your lowest cost implementation. The overall development cost for this approach would be less than the cost of a whole system redesign and the per-unit cost is less than the cost of either external dongle approach.

An Optimized Bridge Solution for Application-Specific USB-to-Serial Applications

To illustrate the flexibility a single USB-to-serial converter provides for any application-specific USB-to-serial application"whether an external dongle implementation or an embedded bridge implementation"TI's TUSB3410 has been used in a PDA docking cradle, a cell-phone MODEM interface, for downloading from a portable health meter and for point-of-sale terminal peripheral upgrades. It contains all the necessary logic to communicate with the host computer using the USB bus. The integrated enhanced UART features include software/hardware flow control including programmable Xon/Xoff characters and programmable auto-RTS/DTR and auto-CTS/DSR. It allows for automatic RS485-bus transceiver control, with and without echo. It has a selectable IrDA mode for up to 115.2 kbps transfer. The UART has a software selectable transfer rate from 50 baud to 921.6 kbaud.

The device enables innovative, flexible solutions via the integrated 8052 microcontroller with 16K bytes of application code space RAM that can be loaded from the PC host or from external on-board memory via an I2C bus. All the device functions such as the USB command decoding, UART setup, and error reporting are managed by the internal MCU firmware under the auspices of the PC host. This firmware-based architecture allows the system developer to address the "operational variations" of the serial ports on the various system processors/microcontrollers. It also offers four general-purpose I/O pins to allow the developer the flexibility to market new and differentiated solutions.

Why new drivers?

The common theme in any of these USB-to-serial bridge approaches is the need for a new virtual COM port driver. This filter driver takes the COM commands generated by the user application and converts them to USB protocol that the bridge device accepts on its USB port. The bridge device is then responsible for converting these USB commands back into serial commands. The driver is responsible for allowing the USB device to appear as a COM port to the operating system (OS).

It allows the user application to ignore that it is actually communicating over a USB connection and presents a standard COM port to the application software. This VCP filter driver would not be required if the designer does not desire to be transparent to the application software and to the system "box." If a change to either of these is acceptable, then the approach (native class drivers) laid out in the system redesign is a potential solution for either of the two application-specific approaches. This approach is not an option for the commercially available dongles.

Texas Instruments offers a product development platform, the TUSB3410UARTPDK. This PDK allows for quick and easy system development of the application-specific bridge. VCP drivers are available for Windows 98/ME and Windows 2000/XP. The application firmware that performs the serial-to-USB conversion is included. The drivers also enable the firmware that is stored on the host to download to the TUSB3410 integrated 8052 RAM. The Header Generator Utility is a DOS-based program that allows the user to generate a header for the EEPROM image file of the application firmware.

More information on the Texas Instruments TUSB3410, as well as TI's other USB solutions, may be found at www.ti.com/usb.

**Ultra-Fast ADC's and Amps
for Any Application**

